MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A150 326

⑤

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR·TR· 85-0018
6a. NAME OF PERFORMING ORGANIZATION Duke University	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c. ADDRESS (City, State and ZIP Code) Dept of Computer Science Durham NC 27706	7b. ADDRESS (City, State and ZIP Code) Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR	8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-84-0132

8c. ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
	61102F	2304	A5	

11. TITLE (Include Security Classification) THE DESIGN OF A UNIFIED PACKAGE FOR THE SOLUTION OF STOCHASTIC PETRI NET MODELS

12. PERSONAL AUTHOR(S) Kishor S. Trivedi, Gianfranco Ciardo, Andrea Bobbio* and Joanne Bechta Dugan

13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM ___ TO ___	14. DATE OF REPORT (Yr., Mo., Day) 1984	15. PAGE COUNT 23

16. SUPPLEMENTARY NOTATION *Istituto Elettrotecnico Nazionale Galileo Ferraris, Torino, Italy.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes the philosophical differences between three current SPN models in an attempt to merge the most important (and non-contradictory) aspects into one. It previews the design of a package for the solution of this unified model.

DTIC
ELECTE
FEB 19 1985
E
D

DTIC FILE COPY

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL MAJ Brian W. Woodruff	22b. TELEPHONE NUMBER (Include Area Code) (202) 767- 5027	22c. OFFICE SYMBOL NM

AFOSR-TR- 85-0013

The Design of a Unified Package

for the Solution of Stochastic Petri Net Models

Joanne Bechta Dugan

Department of Computer Science
Duke University

Andrea Bobbio

Istituto Elettrotecnico Nazionale Galileo Ferraris
Torino, Italy

Gianfranco Ciardo

Department of Computer Science
Duke University

Kishor S. Trivedi

Department of Computer Science
Duke University

ABSTRACT

This paper describes the philosophical differences between three current SPN models
in an attempt to merge the most important (and non-contradictory) aspects into one.
It previews the design of a package for the solution of this unified model.

1. Introduction

Petri net (PN) models were originally intended[1] as a means for a natural representation of

the interaction, logical sequence and synchronization among the elementary activities into which

the operation of a system or the execution of a procedure can be divided. As such, PN's did not convey any information about the duration of each activity or on the way in which the PN-transition to be fired is actually selected from among those enabled in a marking.

Recently, efforts have been made[2, 3, 4, 5, 6, 7] to exploit the modeling power of PN's for representing time-related features of stochastic systems, both in the construction of the model to be solved, and in the definition of the desired output measures. It seems most natural to the authors to associate time with events modeled by PN-transitions; and in fact this choice has been made by the majority of authors in this topic. A Stochastic Petri net is a marked Petri net in which to any PN-transition is associated a random variable, called the *firing time*, characterized by a given distribution function. Moreover, a set of rules are defined for determining the evolution of the net from the knowledge of the firing time distributions. Natkin[2] and Molloy[3] first introduced the SPN with exponentially distributed firing time, so that the stochastic realization of the PN is represented by a continuous-time Markov chain (CTMC).

Moreover, PN's are also useful in modeling interactions between activities which are of a logical nature, and do not correspond to any time-consuming system operation. In order to cope with this problem a generalized class of SPN's (called GSPN's)[5] has been proposed. GSPN's are obtained by allowing PN-transitions to belong to two different classes: immediate PN-transitions and timed PN-transitions. Immediate PN-transitions have higher priority than timed PN-transitions and fire in zero time. A consequence of this partition of PN-transitions is the random switch, which is used to probabilistically determine which among many enabled immediate PN-transitions will fire.

A further generalization of the concept of an SPN was developed[8] where both probabilistic branches (realized by means of a special use of immediate PN-transitions) and general distributions are included. In this general formulation the associated stochastic process in no longer Markovian, and an analytical solution technique has been provided where the firing times have a Phase-type distribution. Furthermore, a numerical technique has been presented[9] for aggregating the state space in the associated Markov chain in the presence of stiffness. In the above men-

tioned line of research a SPN is viewed as a compact way to define a stochastic process isomorphic to the reachability tree of the SPN.

Another somewhat different philosophical view of SPN's is held by the designers of the ESPN (Extended SPN)[6] model and the corresponding solution package DEEP1 (the Duke ESPN Evaluation Package, 1). They see the SPN as a representation of the system being modeled, rather than as a representation of an underlying stochastic process. The ESPN allows an arbitrary probability distribution (even empirical data) for the firing time of a PN-transition. If the ESPN does indeed represent an analytically solvable stochastic process (i.e. a CTMC or a semi-Markov process) it may be solved analytically. [10] If it does not, then the ESPN (not the reachability tree) is simulated. Table 1 presents a comparison between the three SPN models and corresponding solution packages, GSPNA, DEEP1, and IEN-SPN (from the IEN group).

Recently, an effort was begun at Duke University to work toward merging these stochastic Petri net models, both philosophically and physically. The purpose of this paper is to report on this new model and on the design of the corresponding solution package, DEEP. This package will include the most important (non-contradictory) aspects of the three group's works (the GSPN group, the IEN group, and the Duke group), as well as some important additional features. The package will contain both analytic and simulation solution methods, and will address the "stiffness" problem that is often encountered in reliability and performance models. Section two of this paper will describe this new SPN model that we will continue to call an ESPN model. Section three will then discuss the stochastic interpretation of the ESPN. (Here is where the strongest philosophical differences have become apparent.) Then section four will describe the specification and derivation of some queueing, performance and reliability measures on the net. And in section five we will describe the design of the new DEEP, particularly the model specification and model solution methods.

2. The New ESPN Model

The standard Petri net (PN) model [1] is defined by a set of *places*, P, (drawn as circles), a set of *PN-transitions*, T, (drawn as bars), and a set of *directed arcs*, A, which connect PN-transitions to places or places to PN-transitions. Places may contain *tokens*, (drawn as dots). The state of the PN, called the *PN marking*, is defined by a vector enumerating the number of tokens in each place.

A place is an input to a PN-transition if an arc exists from the place to the PN-transition; a place is an output from a PN-transition if an arc exists from the PN-transition to the place. A PN-transition is *enabled* when each of its input places contains at least one token. Enabled PN-transitions can fire, by removing one token from each input place and placing one token in each output place. Thus the firing of a PN-transition causes a change of state (produces a different marking) for the PN. Input and output arcs may have an associated *multiplicity* factor; an arc with a multiplicity factor k is logically equivalent to k arcs. The *reachability tree* is the set of markings that are reachable from a given *initial marking*.

An *inhibitor* arc[1] from a place to a PN-transition has a small circle rather than an arrowhead at the PN-transition. The firing rule for the PN-transition is changed such that the PN-transition is *disabled* if there is at least a token present in the corresponding inhibiting input place. Note that no tokens are moved along inhibitor arcs.

2.1. Modeling Firing Priority

There are a few additional extensions that are included in the ESPN for the sake of user convenience in defining more complex enabling and inhibiting conditions. The concept of arc multiplicity is extended to inhibitor arcs: a multiple inhibitor arc with multiplicity k needs at least k tokens in the corresponding input place to *disable* the PN-transition.

A user may define logical *conditioning* functions for PN-transitions such that a transition is enabled by the normal enabling requirements (including inhibitor arcs) and if the conditioning function is *TRUE*. In the present formulation we allow the user to define a conditioning function

which is a boolean expression possibly involving relational and numerical operators and the number of tokens in various places. Such conditioning functions are often used to reduce the graphical complexity of the Petri net.

Still another way to define inhibitions on the net may be useful to the user. A *priority level* may be defined for each PN-transition in the net, so that the set of PN-transitions is partitioned by priority levels. The firing rule is then modified such that a PN-transition is inhibited if there is a higher priority PN-transition enabled, while the firing of PN-transitions with the same priority level follows the standard rules.

2.2. Modeling Probabilistic Behavior

We may often wish to model the probabilistic outcome of an event represented by the firing of a PN-transition. For example, if a PN-transition represents the running of a diagnostic test on a (possibly) faulty component, its outcomes are *passed* with probability p, and *failed* with probability $1-p$. Thus there would be two output places for the PN-transition, to represent the two distinct outcomes of the one event. We can use a *probabilistic* arc[6] at the output of the PN-transition. (See Figure 1.) In this case the output token would be deposited in the place labeled *passed* with probability p, or the place labeled *failed* with probability $1-p$. A probabilistic arc may have any (finite) number of output places, and may have an associated multiplicity factor, k, in which case all k tokens are deposited in the selected place.

Probabilistic arcs are useful for modeling rather simple probabilistic behavior (typically a branching process) but can become unwieldy for more complex situations. We include a different construct for such situations: the *random switch*[5]. A random switch is defined on a set of PN-transitions, such that when a subset of this set is enabled, a probability of firing (normalized so that they sum to 1) is defined for each enabled PN-transition. The particular PN-transition to fire is then chosen according to this (discrete) probability distribution. If we require that the sets of PN-transitions for each random switch be disjoint, then we can further assign a priority level to each random switch. This prioritizing of random switches defines the outcome when PN-

transitions belonging to two random switches are simultaneously enabled. (Any enabled PN-transition that does not belong to a random switch will be considered a random switch with one element.) The choice of the random switch is made according to the priority level of the switch; the choice of the PN-transition within the chosen random switch is made as if it were the only one active.

It is important to note here that the probabilities for the probabilistic arc and for the random switch need not be constant values; they can each be a function of the current marking. Very complex behavior can be modeled in this way; the complexity is reflected in the way in which the places are combined together (sometimes combinatorial functions are needed). The function could even be expressed in a tabular form, either because the expression of the function is too complex, or because it is not known in a closed form. Both are common cases when a single PN-transition is replacing a complex decision subnet.

2.3. Modeling the Timing of Events

A partition (with respect to timing) is assumed on the set of PN-transitions: each PN-transition is either *immediate* (fires in zero time) or *timed* (fires in an amount of time >0). For each timed PN-transition t_k, the cumulative distribution function $G_k(t \mid M_i)$ must be provided. The parameters of the distribution may depend on the particular marking in which the PN-transition is enabled, hence the distributional dependence on marking M_i. This distribution represents the time needed to complete the activity associated with the PN-transition t_k in marking M_i, given that it is the activity that will complete, and it is independent of the other activities that may occur in the marking.

If more than one PN-transition is enabled for a given marking we need to define a set of rules for choosing the one to fire. If there is an immediate PN-transition among those enabled, it fires first, with probability one. If more than one immediate PN-transition is simultaneously enabled, the one to fire is chosen according to the random switch(es) that contain the PN-transitions.

If there are only timed PN-transitions enabled the most logical choice for the one to fire is that with the minimum firing time. However, there are other scheduling policies the user may wish to enforce, as in *SPADE*.[11] for example a probabilistic selection of one of many competing activities to commence. This selection can be reflected in the net itself, by using a probabilistic arc on an immediate PN-transition. Further, if an activity must wait until all of a set of transitions has fired (effectively the maximum of the firing times), this situation can also be reflected in the net. (See Figure 2.)

3. Stochastic Interpretation

In order to give a stochastic interpretation to the process described by a PN, we should first analyze the impact of the immediate PN-transitions. A marking is called a *vanishing* marking[5] if it enables (at least) one immediate PN-transition. A vanishing marking is so named since no time is spent in this marking; as soon as a vanishing marking is reached (one of) the immediate PN-transition(s) fires in zero time. If a marking enables only timed PN-transitions then it is called a *tangible* marking. Conceptually, a reduction of the reachability tree is possible, by classifying markings into these two classes, and eliminating the vanishing class.

If a marking enables multiple PN-transitions, some of which are immediate, then by definition the PN-transition that will fire is chosen among the immediate PN-transitions only. Thus in the reachability tree the exiting arcs from a marking are either all corresponding to timed PN-transitions (if the marking is tangible) or all corresponding to immediate PN-transitions (if the marking is vanishing). We can then remove the vanishing markings from the reachability tree by reflecting their probabilistic branching into the predecessor tangible markings. This process can be repeated until only tangible markings exist in the reduced reachability tree.

This description applies to the simulation approach as well, in the sense that vanishing marking are treated differently (for example no time counter is changed when leaving a vanishing marking). It must be clear then that the following discussion about timing issues applies to the process described by the tangible markings only.

The definitions introduced so far are not sufficient to completely determine the time-evolution of the net. Further rules have to be assigned, in particular to define how the future behavior is dependent upon the past. Even if the same set of rules can be assigned for both an analytical solution and a simulative solution, the analytical solution requires more stringent assumptions to be numerically tractable.

In the analytical approach we associate a stochastic process with the reachability tree of the PN (tangible states only), such that a discrete state in the stochastic process corresponds to a marking in the PN reachability tree. The nature of the associated stochastic process depends on the policy that is used to keep track of the past at the time in which a state change occurs. We call the set of conditioning rules *state change policies*.

In the simulative approach, instead, we do not generate the reachability tree or simulate the associated stochastic process, but directly simulate the net. It is more natural in this approach to look at the PN-transitions as representing activities whose interactions depend locally on the net structure rather than on the current marking. In particular activities can be concurrent (or non-interacting) or conflicting: in this last case conflict means that one activity is interrupted when some other activity completes. The set of rules for modelling the time dependent behavior of competing activities is called *interruption policies*.

3.1. State change policies

Two different sets of rules have been discussed [8] for associating an analytically defined stochastic process to a generic PN; a numerical procedure has been presented in the case the firing time distributions $G_k(t \mid M_i)$ are of Phase (PH) type. We refer to those rules as *state change restart* and *state change resume* policies.

3.1.1. State change restart

As a state changes any transition enabled in the new marking restarts from scratch. Under the assumption that the firing times are exponentially distributed the associated process becomes a continuous-time homogeneous Markov chain, [12, 5, 13] while in the case of generally distributed

firing times the associated process becomes semi-Markov[2, 6].

It should however be stressed that this policy is not realistic for parallel or concurrent PN-transitions; in fact parallelism implies that each activity runs until completion independent on whether some other activity in some other location of the net has completed. Dugan, et. al[6]. have defined the actual environment of application of this policy, by allowing only exclusive or competing PN-transitions to be generally distributed, the others being exponential.

For simulation purposes, this policy means that we have to generate a new sample for all enabled transitions each time a PN-transition fires, thus forcing a state change. However for the generation of this new sample two different options can be invoked: the *identical sample* and the *different sample* option. (This terminology is borrowed from the preemption of service interruption in queuing theory; [14] it should be emphasized that in the present context a state change does not imply that all the previously enabled activities are interrupted).

The different sample option means that we generate a new deviate for any new enabled PN-transition independently of whether or not it has been disabled by the state change. On the other hand, the identical option means that we maintain the same deviate (but still reset the firing time) for a transition which was already enabled before the state change.

This rather subtle distinction becomes evident if we allow the parameters of the firing time distributions to be marking dependent (even in the exponential case). In the *different sample* option (which is analytically easier to handle)[15], we account for the possibly modified distribution at each state change, while in the *identical sample* policy we account for the possibly modified distribution only when the PN-transition is enabled again after firing (intermediate state changes do not affect the value of the deviate). Rather interesting behavior can be observed using the *different sample restart* policy[16].

3.1.2. State change resume

If we wish to model the more realistic situation in which the enabling of a PN-transition can continue through a change of state, we need to allow the countdown of a firing time to resume in

the new state (marking). This policy is the only reasonable one for modeling the execution of parallel or concurrent activities. It is important to note that the distinction between *state change restart* (with different sample) and *state change resume* is only important when the distributions involved are not exponential, while they are equivalent in the exponential case due to the memoryless property of the exponential distribution. It should also be noted that *state change restart* with identical sample enforces additional memory even in the case of exponentially distributed firing times and hence will produce different results then the *state change resume* policy.

The implementation of the *state change resume* policy requires that each PN-transition t_k keep track of the elapsed time since enabling, through an associated "age variable" a_k. Then, when a state change occurs and the PN-transition is still enabled, the time remaining depends on both the original firing time distribution, and on the age variable, a_k. If the Cdf is not marking dependent, we can write $G_k(t \mid M_i) = G_k(t)$. It follows that at any time t the distribution function of the time to complete the activity associated with the PN-transition t_k is the residual life distribution of t_k given the age a_k, for which the following expression holds:

$$G_k(t \mid a_k) = \frac{G_k(t + a_k) - G_k(a_k)}{1 - G_k(a_k)}$$

In the case of marking dependent distributions this residual life distribution assumes a very complex form since it depends not on a single age variable but on the individual time intervals spent in any previous marking. It is for this reason that we will not allow marking dependencies in this case.

Simulation of the *state change resume* policy at the PN level, implies that a deviate needs to be generated only when a PN transition is first enabled. This deviate is added to the present value of the global clock to produce a firing time. In the case where another transition fires first, without disabling the given transition, this firing time is unchanged.

3.2. Interruption Policies

With the term *interruption* we represent the situation in which a PN-transition was enabled, was disabled for a non-zero amount of time, and is subsequently re-enabled. The question arises as to the firing time of the interrupted PN-transition when it is re-enabled. We have the same policies available as in the previous case: *restart* and *resume* . Unless the firing distributions are all exponential (in which case the *interruption* policy coincides with the *state change* policy), an analytical definition of the stochastic process associated with this policy is not easily envisaged; in fact under this policy we allow a rather complex dependence of the current state of the process on the previous history of the process itself.

3.2.1. Interruption Restart

Again, the simplest answer to the problem of the firing time of a re-enabled interrupted PN-transition, is to assume that the activity restarts from scratch upon re-enabling. This policy is termed *interruption restart* and is certainly valid in a number of cases, most commonly in program execution time[17].

While the analytical approach to this policy may be rather complex, the interruption mechanism is, on the contrary, easily accounted for by simulating the net. Again, the *identical* and the *different* sample options are legal in this case, for the generation of the new sample when an interrupted PN-transition is re-enabled. The *identical sample* option is certainly more valid in the case of an interrupted user-program.

3.2.2. Interruption Resume

An equally valid policy arises, when a re-enabled PN-transition does not waste the time already spent, but resumes the activity at the point of interruption. In this case, as before, we can associate an age variable with each PN-transition.

It is important to note the difference between *state change* policies and *interruption* policies. It is not unreasonable to assign the *state change resume* and *interruption restart* policies

to the same PN-transition. Also note that for firing times that are exponentially distributed, *interruption restart* (with different sample) is stochastically identical to *interruption resume*.

4. Derived Measures of Reliability and Performance

A very important point of the time dependent representation of the system behavior through Petri nets, is that they allow the user to define in a simple and natural way a large number of different measures related to the performance/reliability features of the system.

The output measures are requested to be defined as a part of the input specifications, so that they can be automatically computed during the solution phase both in the analytical and in the simulative approach. Thus the input language is specially structured for providing a friendly environment for the specification of the output measures.

In the analytical approach the knowledge about the system is contained in the time dependent state probability vector which is the basic quantity to be computed. The user defined measures are automatically evaluated as a function of these probabilities.

However, since some of the output measures depend on the integral of the probabilities rather than on the probabilities themselves, it has been found convenient calculating those integrals during the solution step, since they require a small amount of additional computation. In the following discussion we provide final formulas only in the case where the associated stochastic process is a CTMC. In the simulative approach, instead, the solution is directly driven by the output measures specified as an input.

In all the following paragraph it is implicitly intended, unless otherwise specified, that the time t ranges from 0 to ∞, so that all the subsequent discussion applies to the transient as well as to the steady state solution.

4.1. Probability of a Given Condition on the PN

An output condition can be specified as a logical and algebraic function of the number of

tokens in the PN places (e.g. no tokens in the failed places). From the point of view of the input language these output conditions look very similar to the conditioning functions introduced in section 2.1.

In the analytical approach we identify the markings s belonging to the subset C for which the stated condition is true; the output measure $Q_C(t) = Prob\,[the\ condition\ is\ TRUE\ at\ time\ t]$ is given by:

$$Q_C(t) = \sum_{s \in C} P_s(t)$$

where $P_s(t)$ represents the probability of being in state s at time t. If C is the set of operational states, $Q_C(t)$ is the usual definition of reliability (or availability). Additionally, if we define C as the subset of markings in which the system performs at a desired accomplishment level[18], then $Q_C(t)$ can be used in the calculation of the performability at that level.

The estimation of $Q_C(t)$, in the transient simulative solution, is accomplished by quantizing time into small packets of amplitude Δt, and associating a counter with each packet. [19] At each trial the counter number i is incremented by one if at time $i\ \Delta t$ the stated condition is true. At the end of the simulation, each counter is divided by the numbers of trials run, thus providing a discretized frequency estimation of $Q_C(t)$.

A very useful case arises when we want to calculate the transient probability that the condition is satisfied for the first time. By using a standard device in the analysis of stochastic processes, this quantity is evaluated by stopping the process in the subset C in which the condition is fulfilled (we make the states $s \in C$ absorbing), and the calculating $Q_C(t)$ as above.

A similar device is used in simulation where the current trial is stopped as soon as the given condition is realized, and the stopping time is recorded. The $Q_C(t)$ is thus estimated by the frequency of the trials stopped before t.

4.2. Time Spent in a Marking

As in the previous sub-section assume that we are given a particular condition on the net

expressed as a logical function of the number of tokens, and let C be the subset of markings fulfilling that condition. We want to evaluate the time spent in any state s in C during the interval $(0, t)$.

To this end, let us introduce a new random variable Y(t), defined by:

$$Y(t) = \begin{cases} 1 & \text{if the condition is satisfied at time } t \\ 0 & \text{otherwise} \end{cases}$$

Taking the expectation of $Y(t)$, it is easily seen that the expected time spent in C in the interval $(0, t)$ (called $\tau_C(t)$) is given by

$$\tau_C(t) = \sum_{s \in C} \int_0^t P_s(x) dx$$

It is well known from the theory of Markov chains that as t approaches infinity the proportion of time spent in states $s \in C$ equals the asymptotic probability

$$Q_C(\infty) = \sum_{s \in C} P_s(\infty)$$

In the simulative solution the asymptotic probabilities are obtained by the following estimation. At each trial j, a z_j and γ_j are produced in the following way. For each marking change in the trial, a counter z_j is incremented by $Y(t) * (\text{time spent in the marking})$, and a counter γ_j is incremented by the time spent in the marking. After a prescribed number of trials, we set (where n is the number of trials)

$$Q_C(\infty) = \frac{\sum_{j=1}^{n} z_j}{\sum_{j=1}^{n} \gamma_j}$$

Confidence intervals are provided for each estimate as well[10].

4.3. Mean passage time

Given $Q_C(t)$, as calculated in the previous section, is the probability of having entered subset C before t for the first time, the mean first passage time θ_C, has the usual expression:

$$\theta_C = \int_0^\infty (1 - Q_C(t)) dt$$

The above formula requires the transient analysis to be extended over a long interval.

The mean passage time and higher moments can be obtained in simulation; the time to first hitting the given condition is recorded at each trial. The first moment is then estimated by the average value of the clock at the time when the condition is first true, and the second moment is estimated by the average of the squared value of the clock. Confidence intervals for all estimates are provided, for the confidence level desired by the user.

4.4. Cumulative Distribution Function and the Expected Number of Tokens in a Place

Let p_i be a generic place of the PN. The Cdf of the number of tokens in p_i at time t is a staircase function in which the amplitude of the k th step is obtained (from an analytical solution) by summing the probabilities of all the markings containing k tokens ($k = 1,2,3...$) in p_i at time t. From a simulative solution, estimates of the time dependent probabilities of k tokens in the place p_i are required. Obviously the density $f_i(k,t)$ is a mass function equal to the amplitude of the k-th step. The expected value is:

$$E[\, m_i(t)\,] = \sum_{k=0}^{\infty} k \; f_i(k,t)$$

As an example, if place p_i represents processors queuing up for a common resource the above quantities are the Cdf and the expected value of customers in the queue versus time. If p_i represents a failed component, the above quantities are the Cdf and the expected value of the number of failed components at time t.

4.5. Mean Number of Firings of a PN-Transition

Given a time interval $(0 , t)$, this quantity indicates how many times, on the average, an event modeled by a given PN-transition has occurred in that interval. Let t_j be a generic PN-transition, and let C be the subset of the reachable markings which include all the markings $s \in C$ enabling t_j. The mean number of firings of t_j in $(0 , t)$ is given by:

$$n_j(t) = \sum_{s \in C} \int_0^t P_s(z) \lambda_j(s) dz$$

where $\lambda_j(s)$ is the firing rate of t_j in marking s.

In steady state the mean number of firings per unit time becomes:

$$\nu_j = \sum_{s \in C} P_s(\infty) \lambda_j(s)$$

where $P_s(\infty)$ is the steady state probability of state s.

As an example, if t_j indicates the completion of a job, the above quantity is a measure of the number of jobs processed in $(0, t)$ and thus is a measure of the system throughput (job completion rate). Similarly if t_j indicates failure (repair) of a component the above quantity provides the mean number of failures (or repairs) of that component in $(0, t)$.

Again the same quantities can be estimated from the simulative solution. In fact we can record how many times each PN-transition has fired in each trial, and then take averages of these numbers at the end of the simulation.

5. Model Specification and Solution

In this section we will briefly describe the implementation of these ideas in the solution package, *DEEP* (the Duke ESPN evaluation package). There are two interfaces being designed for *DEEP*, one textual, *TIDE* (Textual Interface for *DEEP*), and one graphical, *DIVE* (*DEEP* Interactive Video Editor). When the initial description of the ESPN is entered, it will be stored in a form that either interface can access, so that the user is not "bound" to one type of interface indefinitely. Both are designed with the novice user in mind.

Regardless of which interface is chosen, a "name" can be defined for each place and PN-transition. For each PN-transition, the user defines the input places, inhibiting places (and multiplicities), conditioning functions, and the output places (and multiplicity of output arcs). If a PN-transition is an immediate transition, it is assigned a priority level, and a random switch may be defined. If it is a timed PN-transition, the firing distribution (with possibly marking-dependent parameters), state change and interruption policies (i.e. *restart* or *resume*; *identical* or

different), are also defined. There are default values assigned for the novice user, for example *resume* for both *state change* and *interruption*, and a priority level of zero. The specification of the desired output measures is also part of the input language description.

There are two possible solution methods included in *DEEP*, an analytic Markov chain solver for the solution of the associated stochastic process, and an ESPN simulator to use when the analytic solution is not possible. The user can obtain either a transient or an ergodic solution by either method. The first step in the analytic solution of the ESPN as defined in this paper is the generation of the reachability tree, the set of all markings that are reachable from a given initial marking.

If all the timed PN-transitions are exponentially distributed then the resulting reachability tree is classified as a Markov chain, and solved analytically. If a PN-transition has a Phase-type distribution, an expansion of the state space is necessary before analytic solution. This poses some additional "bookkeeping" problems, since a particular marking in the ESPN is now represented by a cluster of states in the associated stochastic process.

The resulting Markov chain is then solved numerically, for either a transient or an ergodic solution. The solution of the chain provides the (possibly time-dependent) marking probabilities, which must then be transformed into the desired measures on the net by means of the formulas presented in the previous section.

There are many instances in which the stochastic process associated with a particular ESPN is analytically tractable with difficulty at best. These cases include those in which a firing time is not exponentially or Phase-type distributed, or in which the *restart-identical-sample* policy is chosen. In these cases, we resort to the simulation of the ESPN for solution. There are two types of ESPN simulation available, a transient simulation and an ergodic simulation, both of which gather statistics from many sample paths through the net. We are currently studying ways to make the simulation more efficient, especially in the case of simulating "stiff" systems or rare events.

6. Conclusions

We have presented the views of three research groups on the theory and solution of stochastic Petri nets, in an attempt to merge them into one. The results of this integration of both models and solution techniques are embodied in the design of a new ESPN model and corresponding solution package. We believe that the ESPN model as defined in this paper represents a unification of the most important aspects of the three separate models. We also feel that the solution package will gain from both our individual experiences with SPN's, and from the many discussions that preceded the writing of this paper.

7. Acknowledgements

This effort was begun while Andrea Bobbio was a visiting scientist at Duke University, in the Fall semester, 1984. We also wish to acknowledge the other members of the research groups: from the IEN group, Aldo Cumani; from the GSPN group, Marco Ajmone Marsan, Gianfranco Balbo and Giovanni Conte; and from Duke, Victor Nicola, Philip Thambidurai, John White and Roger Smith.

References

1. James L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1981 .

2. S. Natkin, *Reseaux de Petri Stochastiques*, These de Docteur Ingeneur CNAM-Paris, June 1980.

3. Michael K. Molloy, *On the Integration of Delay and Throughput Measures in Distributed Processing Models*, Ph.D. Dissertation, UCLA, 1981.

4. Joseph Sifakis, "Use of Petri Nets for Performance Evaluation," in *Measuring, Modeling and Evaluating Computer Systems*, ed. H. Beilner and E. Gelenbe, pp. 75-91, Elsevier Science Publishers, North-Holland, 1977.

5. M. Ajmone Marsan, Gianfranco Balbo, and Gianni Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Transactions on Computer Systems*, May, 1984.

6. Joanne Bechta Dugan, Kishor S. Trivedi, Robert Geist, and Victor F. Nicola, "Extended Stochastic Petri Nets: Applications and Analysis," *Performance 84*, December 1984.

7. M. Ajmone Marsan, A. Bobbio, G. Conte, and A. Cumani, "Performance analysis of degradable multiprocessor systems using generalized stochastic Petri nets," *IEEE Comp Soc, Distributed Processing T.C. Newsletter*, vol. 6, no. SI-1, pp. 47-54 , 1984.

8. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Conte, and A. Cumani, *Stochastic Petri Nets with Phase-Type Firing Distributions*, November 1984. Paper in Progress

9. A. Bobbio and A. Cumani, "Reduced markovian representation of stochastic Petri net models," *Systems Science*, 1984. To appear.

10. Joanne Bechta Dugan, *Extended Stochastic Petri Nets: Applications and Analysis*, Ph.D. Dissertation, Department of Electrical Engineering, Duke University, 1984.

11. Robin Sahner and Kishor Trivedi, "SPADE: Series-Parallel Directed Acyclic Graph Evaluator," Technical Report CS-1984-15, Department of Computer Science, Duke University, 1984. Submitted for Publication

12. Michael K. Molloy, "Performance Analysis using Stochastic Petri Nets," *IEEE Transactions on Computers*, September, 1982.

13. M. Ajmone Marsan, Gianfranco Balbo, Gianfranco Ciardo, and Gianni Conte, "A Software Tool for the Automatic Analysis of Generalized Stochastic Petri Net Models," in *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis*, Paris, May, 1984.

14. D. P. Gaver, "A Waiting Line with Interrupted Service, Including Priorities," *Journal of the Royal Statistical Society*, pp. 73-90, 1962.

15. Francois Baccelli and Kishor Trivedi, "Analysis of M/G/2 Standby Redundant System," in *Proceedings IFIP Symposium, PERFORMANCE '83*, College Park, Md.

16. Kishor Trivedi, "Modeling and Analysis of Fault-Tolerant Systems," in *Proceedings of the International Conference on Modeling Techniques and Tools for Performance Analysis*, Paris, May 1984.

17. Vidyadhar Kulkarni, Victor Nicola, and Kishor Trivedi, *On Modeling the Performance and Reliability of Multi-mode Computer Systems*, 1984. submitted for publication.

18. John F. Meyer, "Closed-Form Solutions of Performability," *IEEE Transactions on Computers*, pp. 648-657, July, 1982.

19. Robert Geist, Kishor Trivedi, Joanne Bechta Dugan, and Mark Smotherman, "Modeling Imperfect Coverage in Fault-Tolerant Systems," in *Proceedings IEEE 14-th Fault Tolerant Computing Symposium*, pp. 77-82, June, 1984.

TABLE 1

Comparison of the three different packages

DEEP1	GSPNA	IEN-SPN	Features
YES	YES	YES	Multiple arcs
YES	YES	NO	Immediate transitions
YES	NO (1)	YES	Inhibitor arcs
NO	NO (1)	NO	Multiple inhibitor arcs
NO	NO (1)	NO	Marking dependent inhibitions
NO (2)	NO (2)	YES	Priority levels
NO (3)	YES	NO (3)	Random switch constructs
YES	NO (4)	NO (3)	Probabilistic arcs
NO	YES	NO (3)	Marking dependent probabilities
			Firing distributions
YES	YES	YES	exponential
NO (5)	NO	YES	coxian
YES	NO	NO	other
YES	YES	YES	Marking dependent parameters
			Solution method
SIMULATION (6)	ANALYTICAL	ANALYTICAL	steady state
SIMULATION (6)	NO	ANALYTICAL	transient
YES	NO	NO	Graphical input capability
YES	YES	YES	Textual input capability
NO	YES	NO	Textual editor

NOTES
(1) simulated by setting a rate or a probability to 0
(2) implicit priority of immediate transitions over timed ones
(3) simulated with high rate transitions
(4) obtained as a particular case of the random switch construct
(5) for the simulation, ERLANG, HYPEREXP, HYPOEXP are predefined
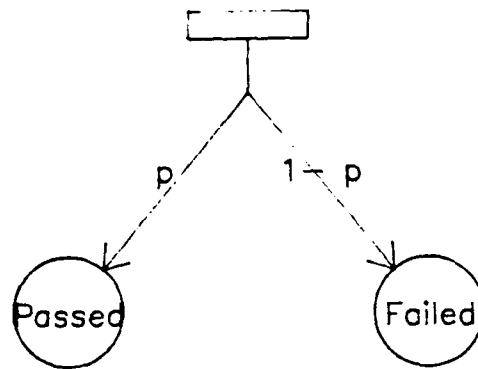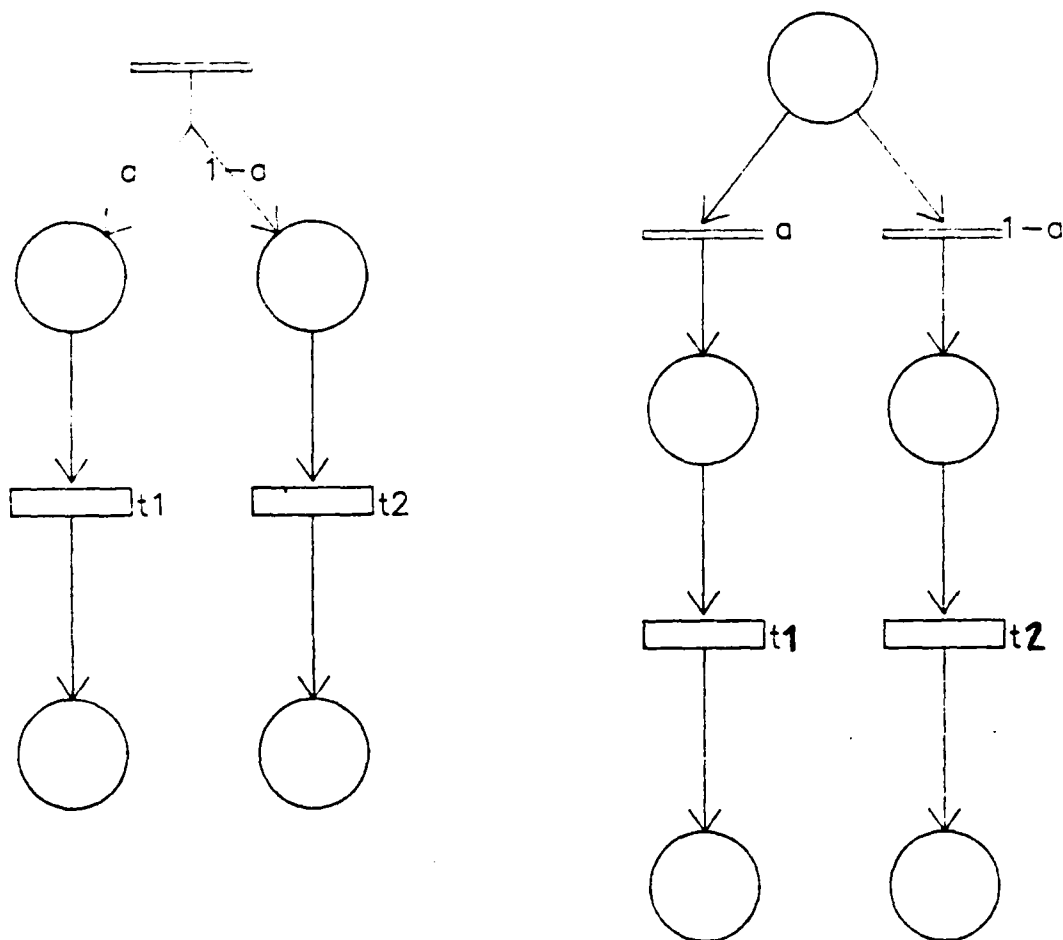(6) analytic solution of some cases is included in the design

Figure 1. A Probabilistic Arc
When the transition fires, a token is deposited in the place labeled *passed* with probability p and in the place labeled *failed* with probability $1-p$.

a) Pre-Selection: The activity represented by transition t_1 is selected with probability a; that represented by transition t_2 is selected with probability $1-a$.

Figure 2. Implementation of Different Scheduling Policies
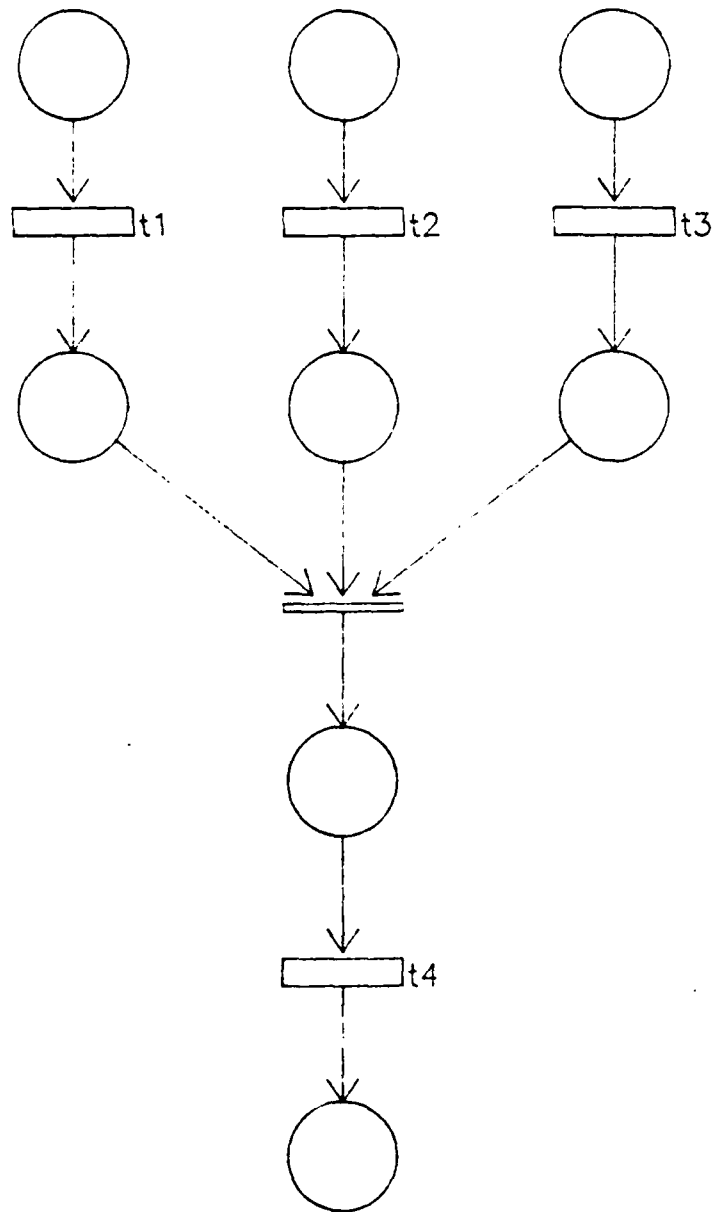
b) Maximum: The activity represented by transition t_4 must wait until transitions t_1, t_2 and t_3 complete.

Figure 2. Implementation of Different Scheduling Policies

END

FILMED

3-85

DTIC